2. Ecommerce Platform Search Function

Asymptotic Notation and Algorithm Analysis:

- **Big O Notation**:
  - Big O notation (often denoted as O(n)) describes the upper bound of an algorithm's running time as the input size grows.
  - It helps us analyze how an algorithm performs in terms of time complexity.
  - For example, when we say an algorithm has a time complexity of O(n), it means that its execution time grows linearly with the input size.
  - Big O notation is essential for understanding worst-case scenarios.
- **Best, Average, and Worst-Case Scenarios for Search Operations**:
  - In search algorithms:
    - **Best Case**: Represents the minimum time required for a successful search (e.g., finding the desired item on the first try).
    - **Average Case**: Reflects the expected time considering various inputs (e.g., random data distribution).
    - **Worst Case**: Indicates the maximum time needed for a search (e.g., when the desired item is at the end of the list or not present).
  - Linear search and binary search exhibit different behaviors:
    - **Linear Search**:
      - Best Case: O(1) (when the target item is the first element).
      - Average Case: O(n/2) (assuming random data distribution).
      - Worst Case: O(n) (when the target item is at the end or not found).
    - **Binary Search**:
      - Best Case: O(1) (when the target item is the middle element).
      - Average Case: O(log n) (assuming a sorted array).
      - Worst Case: O(log n) (same as average case).
  - Binary search significantly outperforms linear search for large datasets due to its logarithmic time complexity.

Implementation and Platform Suitability:

- **Product Class Attributes**:
  - Create a Product class with attributes like productId, productName, and category.

- **Linear Search**:
    - Linear search iterates through each element in the array until it finds the desired item or reaches the end. It’s simple but inefficient for large datasets.
    - Simple to implement but inefficient for large inventories.
    - Suitable for small datasets or unsorted lists.
    - Store products in an array and iterate through each element.
    - Best Case: O(1) (when the target item is the first element).
    - Average Case: O(n/2) (assuming random data distribution).
    - Worst Case: O(n) (when the target item is at the end or not found).
- **Binary Search**:
    - Binary search divides the array in half at each step, narrowing down the search range. It’s highly efficient for sorted arrays and significantly outperforms linear search for large inventories.
    - Requires a sorted array.
    - Efficient for large inventories.
    - Divide and conquer approach.
    - Choose binary search if your platform deals with substantial product catalogs.
    - Consider using a balanced binary search tree (e.g., AVL tree) for even better performance.
    - Best Case: O(1) (when the target item is the middle element).
    - Average Case: O(log n) (assuming a sorted array).
    - Worst Case: O(log n) (same as average case).